André M. Santamaría Regal

DNI: 36117001D

Enunciado

1.- Documenta el proceso desde la creación de una cuenta gratuita Neo4J AuraDB en ***https://neo4j.com/cloud/platform/aura-graph-database/*** hasta tener operativa una instancia de base de datos. Captura las pantallas donde sea necesaria alguna interacción por parte del usuario y justifica las opciones que vas eligiendo.

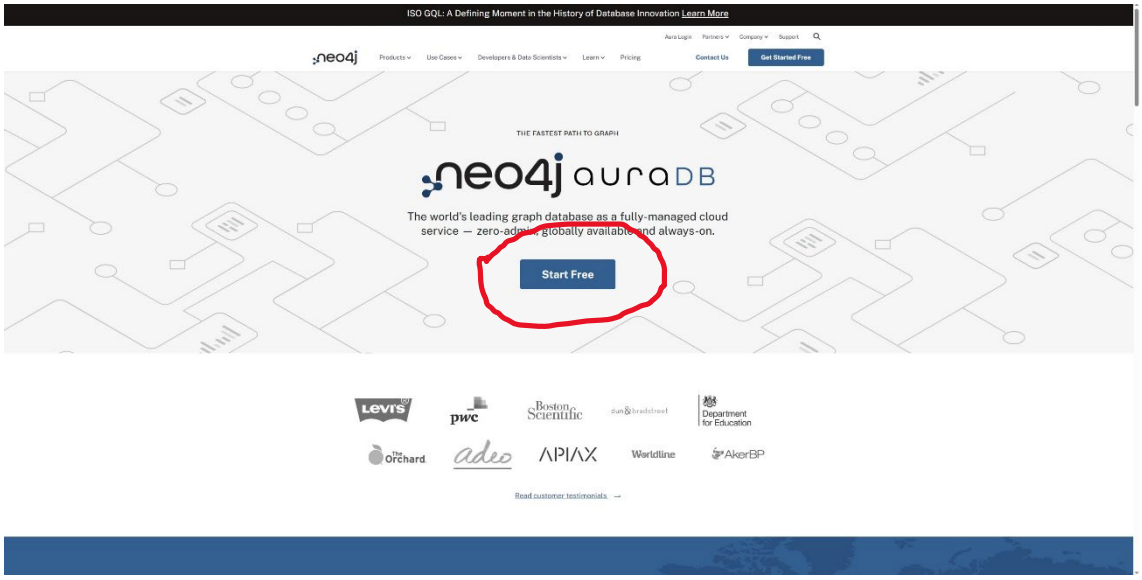


Imagen 1

En esta imagen 1 se muestra la landing page de la web de Neo4j Aura DB, donde tenemos que crear un usuario para poder acceder a las diferentes opciones que nos ofrece, empezaremos haciendo click en start free para proceder a dicha creación de usuario.

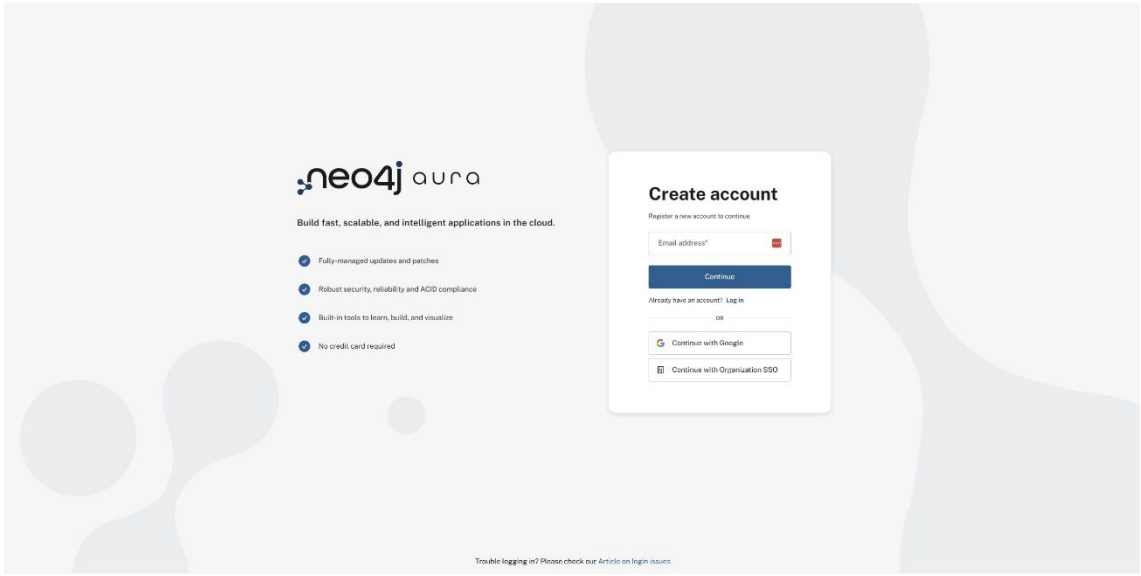


Imagen 2

En la imagen 2 podemos observar las opciones de creación de cuenta (Create account), bien sea por medio del email, con una cuenta de google o por medio de una ID empresarial.

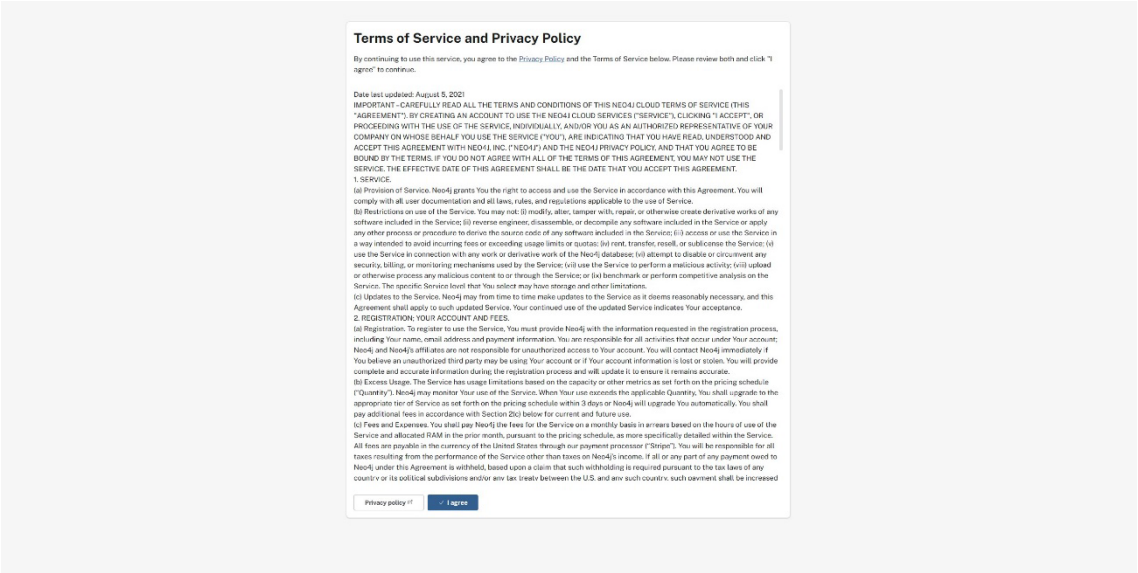


Imagen 3

La imagen 3 muestra los terminos del servicio y su política de privacidad para que aceptes dichos términos para poder continuar con la creación de la cuenta de usuario.

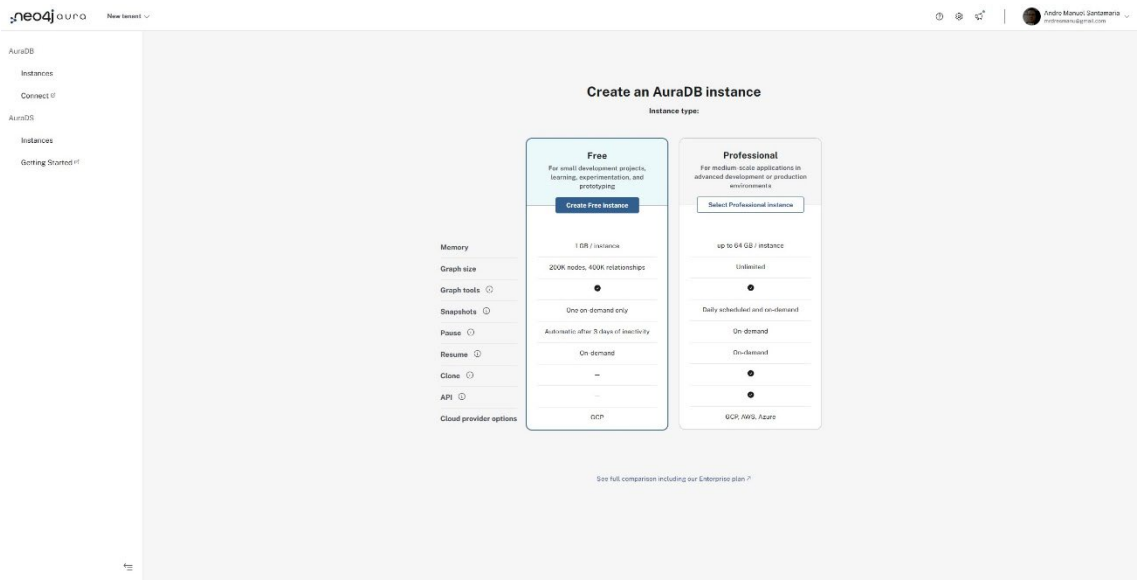


Imagen 4

Imagen 4 donde se muestra las opciones de instancias de AuraDB, en nuestro caso crearemos una instancia gratuita.

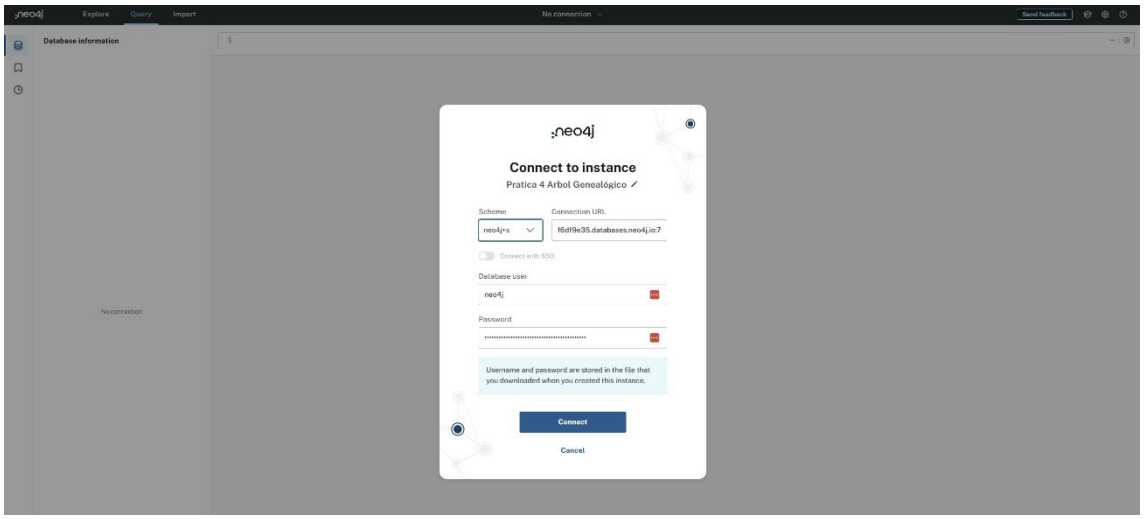
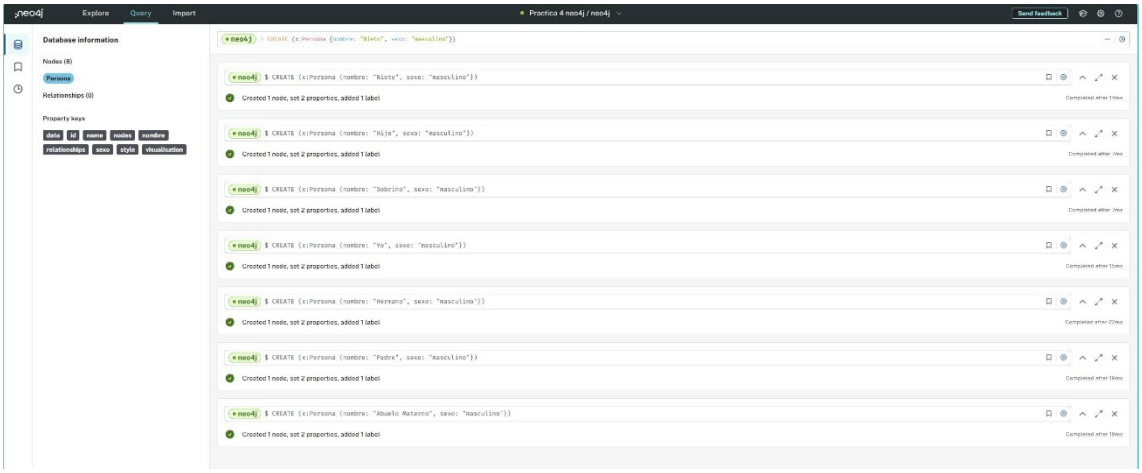


Imagen 5

En la imagen 5 se puede ver la creación y conexión con la instancia de la base de datos, es necesario el tener un usuario y un password de base de datos.

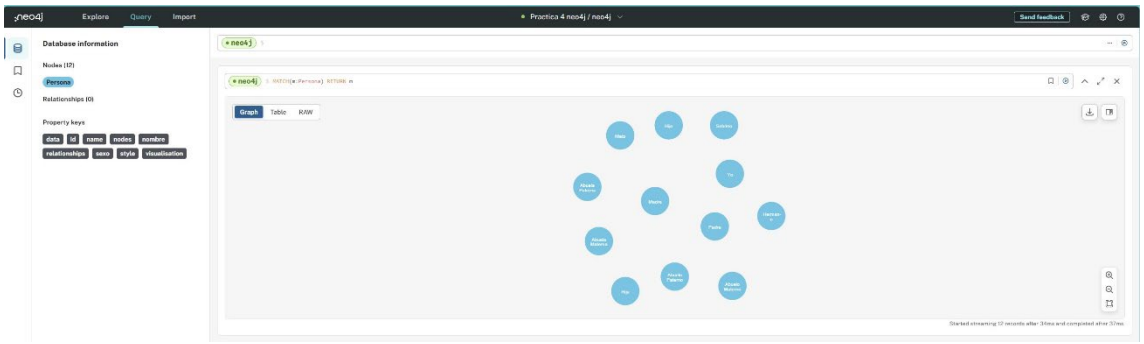
2.- Crea los 12 nodos necesarios para representar la siguiente imagen:

Cada nodo debe estar etiquetado como “Persona” y tener las propiedades nombre y sexo con el valor que corresponda en cada caso.
Captura la pantalla en la que se vea el/los comandos que utilizas para crear alguno de estos nodos.
Explica con tus palabras cada parte del comando y sus parámetros.



Para crear un nodo con las características requeridas se utiliza el comando que se puede apreciar en las imágenes: CREATE (x:Persona { nombre:“XXXXXX” , sexo:”YYYYYYY”})
CREATE es el comando usado para crear un nodo en la DB, para ello entre paréntesis ponemos un nombre de variable en este caso X y un valor después de los : que indica el tipo del nodo, en este caso el tipo es Persona, luego entre llaves le damos las propiedades al nodo que queramos, en este caso serian nombre y sexo a los que después del : asignaremos el valor correspondiente.

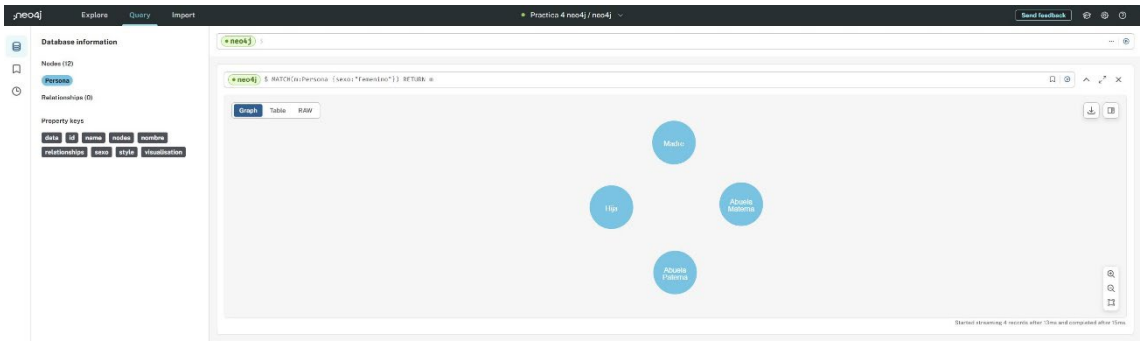
3.- Muestra un gráfico a través de Neo4J con todos los nodos que has creado. Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.



El comando utilizado para listar los nodos creados en el paso anterior es el comando

MATCH (m:Persona) RETURN m , a este comando se le pide que en una variable m se guarden todos los nodos Persona de la DB y luego con el comando RETURN que nos devuelva lo almacenado en dicha variable.

4.- Muestra un gráfico a través de Neo4J con todos los nodos de sexo “Mujer” que has creado. Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.



El comando utilizado es:
MATCH(m:Persona {sexo:”femenino”}) RETURN m, con el comando MATCH se pide que se almacenen en la variable m todos los nodos de tipo Persona cuya propiedad sexo tenga un valor “femenino” y luego con el comando RETURN se devuelve dicha variable y se muestra gráficamente.

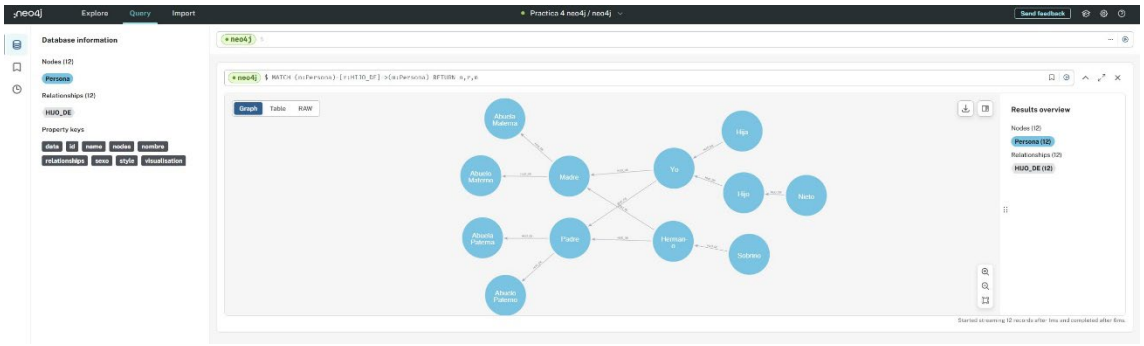
5.- Crea las relaciones “HIJO_DE” que se ven en la imagen:

Captura la pantalla en la que se vea el/los comandos que utilizas para crear alguna de estas relaciones. Explica con tus palabras cada parte del comando y sus parámetros.

MATCH (p:Persona{nombre:"Madre"}) , (x:Persona{nombre:"Abuela Materna"}) CREATE (p) - [:HIJO_DE]->(x)

Con el comando MATCH buscamos los dos nodos Persona que queremos relacionar, en este caso los que en su propiedad nombre tienen el valor Madre y Abuela Materna, una vez cargados en su respectiva variable usando el comando CREATE creamos la relación HIJO_DE que es la parte que va entre corchetes y con la flecha -> indicamos el sentido hacia que nodo va la relación.

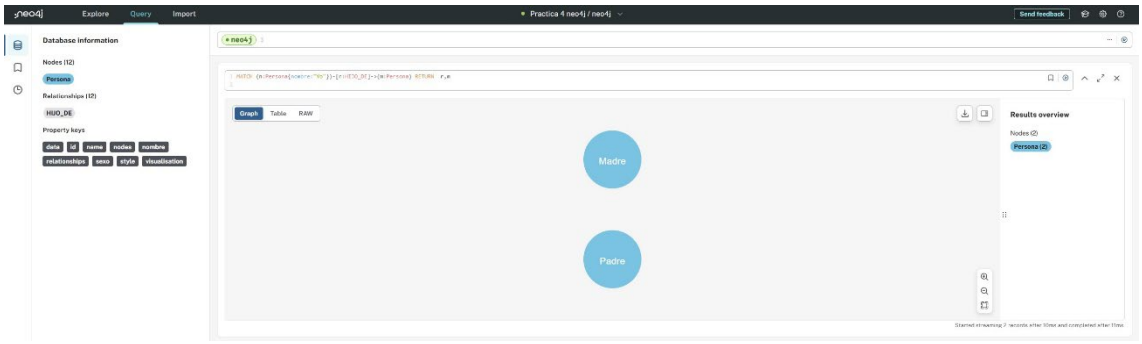
6.- Muestra un gráfico a través de Neo4J en el que se vean todos los nodos y relaciones que has creado. Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.



MATCH (n:Persona)-[r:HIJO_DE]->(m:Persona) RETURN n,r,m

Con este comando MATCH buscamos los nodos Persona (n:Persona) que tiene la relación [r:Hijo_D] con otro nodo Persona (m:Persona) y los devolvemos con RETURN n,r,m

7.- Muestra un gráfico en el que solo se vean los ascendentes directos de un determinado nodo, por ejemplo si buscamos para el nodo “Yo” debería aparecer los nodos “Padre” y “Madre”. Si buscamos para el nodo “Hijo” solo debo aparecer “Yo”.
Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.



MATCH (n:Persona{nombre:"Yo"})-[r:HIJO_DE]->(m:Persona) RETURN r,m

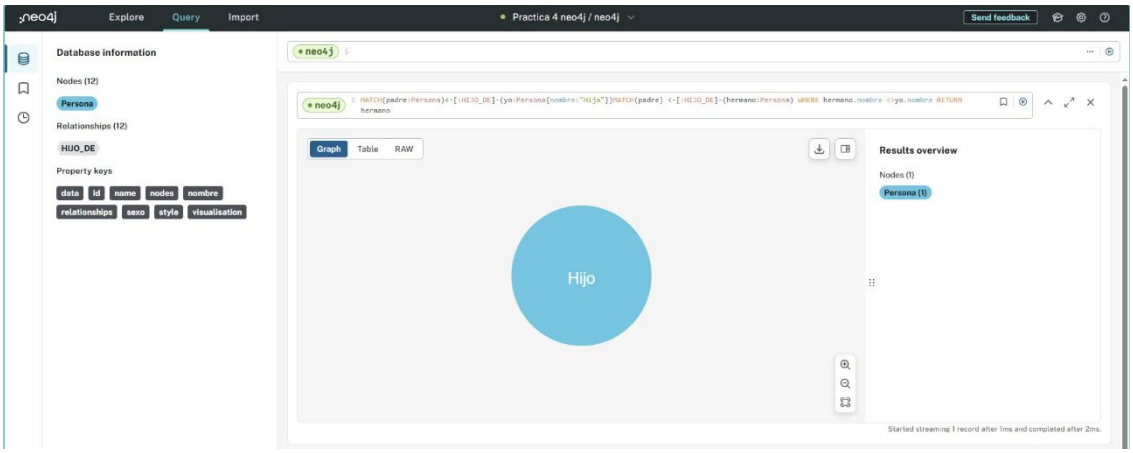
Con match buscamos el nodo Persona cuya propiedad nombre tenga el valor elegido (Yo) y con una relación HIJO_DE en dirección al/los nodo/s m:Persona y para visualizar lo requerido utilizamos el comando RETURN sin incluir la variable que almacena el nodo del que queremos saber sus ascendientes.

8.- Muestra un gráfico en el que se vean los segundos ascendientes directos masculinos (los abuelos) de un determinado nodo. Por ejemplo, aplicado al nodo “Yo” debería aparecer “Abuelo Materno” y Abuelo Paterno”. Aplicado al nodo “Hijo” solo debería aparecer “Padre”. Aplicado a “Padre” no debería aparecer nada.
Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.

MATCH (n:Persona{nombre:"Yo"})-[r:HIJO_DE]->(m:Persona)-[h:HIJO_DE]->(j:Persona{sexo:"masculino"}) RETURN j

Con MATCH buscamos el nodo Persona cuyo nombre sea del que queremos saber los segundos ascendientes masculinos, este tiene una relación HIJO_DE con los nodos m:Persona que serian sus primeros ascendientes y a su vez estos tienen relación HIJO_DE con los nodos j:Persona que serían sus segundos ascendientes y como solo buscamos los masculinos necesitamos almacenar solo los que tienen en su propiedad sexo el valor masculino, posteriormente con el comando RETURN mostramos solo lo almacena en j ya que estos son como dije antes los segundos ascendientes masculinos de n.

8.- Muestra un gráfico en el que se vean los hermanos de un nodo (los que comparten algún ascendiente directo). Por ejemplo, aplicado al nodo “Yo” debería aparecer el nodo “Hermano”, aplicado a “Hija” debería aparecer “Hijo”, aplicado a “Nieto” no debería aparecer nada y aplicado a “Madre” tampoco debería aparecer nada.
Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.



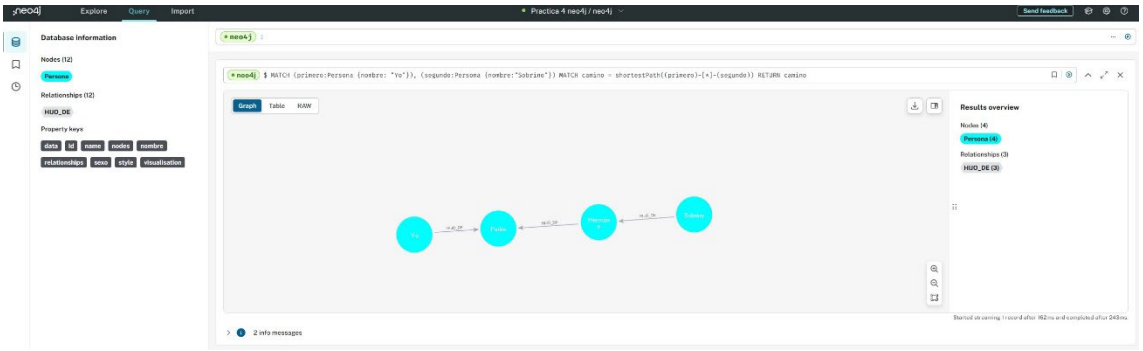
MATCH(padre:Persona)<-[:HIJO_DE]-(yo:Persona{nombre:"Hija"})MATCH(padre) <-[:HIJO_DE]-(hermano:Persona) WHERE hermano.nombre <>yo.nombre RETURN hermano

En esta primera parte de la consulta MATCH(padre:Persona)<-[:HIJO_DE]-(yo:Persona{nombre:"Hija"}) los que se hace es almacenar en la variable padre el nodo persona que es nodo padre del nodo almacenado en la variable yo.
MATCH(padre) <-[:HIJO_DE]-(hermano:Persona) WHERE hermano.nombre <>yo.nombre RETURN hermano en esta segunda parte de la consulta lo que hacemos es utilizar el nodo padre para almacenar en la variable hermano todos los nodos que tienen al nodo padre por padre filtrando con el comando WHERE para que no muestre lo almacenado en la variable yo.

9.- Muestra un gráfico en el que se vean los tíos de un nodo. Por ejemplo, aplicado al nodo “Sobrino” debería aparecer yo. Aplicado a “Hijo” debería aparecer “Hermano” y aplicado a “Yo” no debería aparecer nada.
Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.

MATCH (hijo:Persona {nombre: "Sobrino"})-[:HIJO_DE]->(padre) MATCH (padre)-[:HIJO_DE]->(abuelo) MATCH (abuelo)<-[:HIJO_DE]-(tio:Persona)WHERE tio.nombre <> padre.nombre RETURN tio
En la primera parte de la consulta buscamos el nodo padre del nodo al que queremos encontrar su tio, luego subimos un nivel mas al nodo abuelo para posteriormente buscar los nodos hijos de este nodo abuelo y mostramos el resultado quitando el nodo padre.

10.-Muestra un gráfico en el que se vea la ruta más corta entre dos nodos cualesquiera. Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.

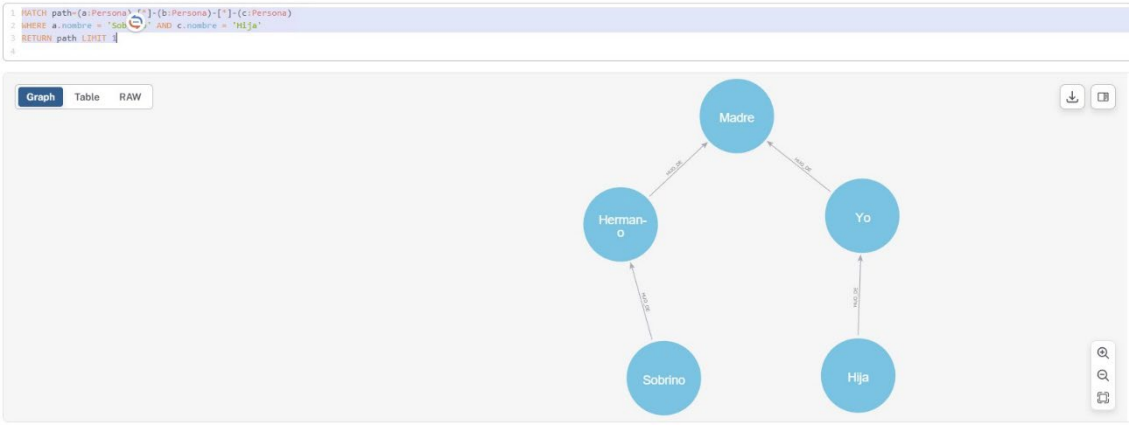


MATCH (primero:Persona {nombre: "Yo"}), (segundo:Persona {nombre:"Sobrino"}) MATCH camino = shortestPath((primero)-[*]-(segundo)) RETURN camino

Empezamos poniendo en las variables primero y segundo los nodos de los que queremos encontrar el camino mas corto y luego con el comando shortestPath se almacena en la variable camino los saltos entre nodos que hacen el camino mas corto entre los dos nodos y posteriormente con Return mostramos lo almacenado en la variable camino.

11.-Muestra un gráfico en el que se vea el primer nodo en común entre dos nodos cualesquiera. Captura la pantalla donde se vea el comando que utilizas y el gráfico resultante. Explica con tus palabras los comandos usados y sus parámetros.

MATCH path=(a:Persona)-[*]-(b:Persona)-[*]-(c:Persona) WHERE a.nombre = 'Sobrino' AND c.nombre = 'Hija' RETURN path LIMIT 1



Después de darle muchas vueltas y pruebas no he sido capaz de encontrar la consulta exacta que se pide, esto se acerca en cuanto saca el camino en común entre los dos nodos pero no saca un único nodo.